FASI DELLA PROGETTAZIONE DI UNA BASE DI DATI

La progettazione di una base di dati si divide generalmente in tre fasi principali:

1) PROGETTAZIONE CONCETTUALE DI UN DATABASE

La **progettazione concettuale di un database** è la fase iniziale dello sviluppo di un database, in cui si definisce la struttura logica dei dati senza considerare dettagli tecnici specifici del sistema di gestione del database (DBMS). L'obiettivo è creare un modello chiaro e comprensibile dei dati e delle loro relazioni. Ecco le principali fasi della progettazione concettuale:

- **Analisi dei requisiti**
  In questa fase si definisce il modello concettuale della base di dati. Si identificano le esigenze degli utenti e i processi aziendali.
- **Identificazione delle entità**
  Le **entità** rappresentano oggetti o concetti di interesse (es. Cliente, Prodotto, Ordine).
- **Definizione delle relazioni**
  Determinare le **relazioni** tra le entità (es. un Cliente effettua uno o più Ordini).
  Specificare il **cardinalità** delle relazioni (1:1, 1:N, N:M).
- **Creazione del Modello E-R (Entity-Relationship)**

Si utilizza il **diagramma E-R** per rappresentare graficamente le entità, le relazioni e gli attributi.
Si definiscono le chiavi primarie e le chiavi esterne.
Si identificano le gerarchie (specializzazione/generalizzazione).

Caratteristiche Principali Modello E-R

1. Entità

- Un'**entità** rappresenta un oggetto concreto o astratto della realtà che deve essere memorizzato nel database.
- Può essere un **oggetto fisico** (es. Studente, Libro, Automobile) o un **concetto** (es. Corso, Ordine).

Tipi di entità

- **Entità forte** → Ha un'esistenza autonoma e una **chiave primaria** che la identifica univocamente (es. Studente con ID).
- **Entità debole** → Dipende da un'altra entità e non ha una chiave propria (es. Fattura che dipende da un Cliente).

2. Attributi

- Gli **attributi** sono le caratteristiche o proprietà di un'entità.
- Ogni entità possiede uno o più attributi.

Tipi di attributi

- **Semplice** → Non può essere suddiviso (es. Nome, Cognome).
- **Composto** → Può essere diviso in sotto-attributi (es. Indirizzo → Via, Numero Civico, Città).
- **Multivalore** → Può avere più valori per un singolo oggetto (es. un **Docente** può avere più **Numeri di Telefono**).
- **Derivato** → Può essere calcolato da altri attributi (es. l'età può essere derivata dalla data di nascita).
- **Chiave primaria** → Attributo che identifica univocamente un'istanza dell'entità (es. Matricola per Studente).

3. Relazioni

- Una **relazione** collega due o più entità.
- Definisce **come gli oggetti sono connessi** tra loro nel database.

Tipi di relazioni:

- Uno-a-Uno (1:1) → Un entità è associata a una sola istanza dell'altra entità (es. Persona - Codice Fiscale**).**
- Uno-a-Molti (1:N) → Un'entità è collegata a più istanze di un'altra entità (es. Professore - Studenti**).**
- Molti-a-Molti (N:N) → Un'entità può essere associata a più istanze dell'altra e viceversa (es. **Studenti - Corsi**).

4. Cardinalità

- La **cardinalità** specifica **quante istanze di un'entità** possono essere collegate a un'istanza di un'altra entità.

Tipi di cardinalità:

- **(0,1)** → L'istanza può esistere senza relazioni o avere solo un legame (es. una Persona **può** avere un Passaporto, ma non è obbligatorio).
- **(1,1)** → Ogni entità deve avere una relazione fissa con un'altra entità (es. ogni Persona **ha** un solo Codice Fiscale).
- **(0,N)** → L'entità può essere collegata a più istanze, ma non è obbligatorio (es. un Cliente **può** effettuare più Ordini, ma non necessariamente).
- **(1,N)** → Un'entità **deve** essere collegata ad almeno un'altra (es. ogni Studente **deve essere iscritto** ad almeno un Corso).

5. Associazioni con Attributi

- A volte una relazione ha proprietà proprie, che non appartengono a nessuna delle due entità.
- In questi casi, si aggiungono attributi alla relazione stessa.

6. Identificatori Esterni

- Se un'entità non può essere identificata senza riferirsi ad un'altra entità, si usa un identificatore esterno.

7. Generalizzazione e Specializzazione

- La **generalizzazione** rappresenta una relazione **"genitore-figlio"** tra entità.
- Un'entità più generale (**Superclasse**) può essere suddivisa in entità più specifiche (**Sottoclassi**).

Vantaggi del Modello ER

Facilita la progettazione di database in modo chiaro e strutturato.
Indipendente dalla tecnologia di implementazione.
Comprensibile anche a utenti non tecnici.
Facilita l'ottimizzazione del database prima dell'implementazione.

Svantaggi del Modello ER

Può diventare complesso in sistemi molto grandi.
Non rappresenta direttamente i processi aziendali, ma solo i dati.

Esempio: Supponiamo di progettare una base di dati per un negozio online. Le entità principali potrebbero essere:

- Cliente (con attributi come nome, email, indirizzo)
- Prodotto (con attributi come nome, prezzo, categoria)
- Ordine (con attributi come data, stato)

Le relazioni tra le entità potrebbero essere:

- Un cliente può fare molti ordini (relazione tra Cliente e Ordine).
- Un ordine può contenere più prodotti (relazione tra Ordine e Prodotto).

2) PROGETTAZIONE LOGICA DI UN DATABASE

La **progettazione logica di un database** è la fase in cui il modello concettuale viene trasformato in una rappresentazione più dettagliata, adatta alla successiva implementazione in un sistema di gestione di database (DBMS). In questa fase si definiscono le tabelle, le chiavi e i vincoli di integrità, mantenendo un livello astratto che prescinde dal DBMS specifico.

Fasi della progettazione logica

1. Trasformazione del Modello E-R in Modello Relazionale

- **Entità → Tabelle**: ogni entità diventa una tabella con i suoi attributi.
- **Attributi → Colonne**: gli attributi dell'entità diventano colonne della tabella.
- **Relazioni → Chiavi esterne**: le relazioni tra entità vengono rappresentate con chiavi esterne.

2. Definizione delle chiavi

- **Chiave primaria (Primary Key - PK)**: identificatore univoco di ogni record.
- **Chiavi esterne (Foreign Key - FK)**: riferimenti ad altre tabelle per mantenere le relazioni.
- **Chiavi univoche (Unique Key)**: vincoli che impediscono la duplicazione di determinati valori

3. Normalizzazione

La normalizzazione è un processo che serve a ridurre la ridondanza dei dati e a evitare problemi di inconsistenza. Si suddividono le informazioni in più tabelle, con lo scopo di mantenere solo i dati necessari in ogni tabella. Si applicano le **forme normali** per eliminare ridondanze e anomalie.
1NF (Prima Forma Normale): eliminazione di gruppi ripetuti e creazione di tabelle con attributi atomici.

- Tutti gli **attributi devono contenere valori atomici** (non divisibili).
- Non devono esistere **gruppi ripetuti** o **valori multipli** in una colonna.
- Ogni riga deve avere un **identificatore univoco** (chiave primaria).

2NF (Seconda Forma Normale): eliminazione delle dipendenze parziali.

- Deve essere in **1NF**.
- Gli **attributi non chiave** devono dipendere **completamente** dalla chiave primaria.

3NF (Terza Forma Normale): eliminazione delle dipendenze transitive.

- Deve essere in **2NF**.
- Gli **attributi non chiave** devono dipendere **solo dalla chiave primaria** e non da altri attributi non chiave.

Esempio: Invece di memorizzare direttamente i dati dell'indirizzo del cliente in ogni ordine (causando ridondanza), si può creare una tabella separata per gli indirizzi, associata al cliente.
La normalizzazione è fondamentale per garantire **efficienza, coerenza e scalabilità** in un database relazionale. Tuttavia, in alcuni casi può aumentare la complessità delle query a causa del numero maggiore di tabelle e delle relazioni tra esse.

4. Definizione dei Vincoli di Integrità

- **Integrità referenziale**: garantisce la coerenza tra chiavi primarie e chiavi esterne.

- **Integrità dei dati**: impone regole sui valori accettabili nei campi (es. NOT NULL, CHECK).

5. Ottimizzazione del modello logico

- Analisi delle prestazioni e riduzione della complessità.
- Creazione di **indici** per velocizzare le ricerche.
- Uso di viste per migliorare la gestione dei dati.

3) PROGETTAZIONE FISICA DI UN DATABASE

La **progettazione fisica di un database** è la fase finale dello sviluppo del database, in cui il modello logico viene implementato in un sistema di gestione di database (DBMS) specifico. In questa fase si ottimizzano le prestazioni e si gestiscono aspetti fisici come l'allocazione dello spazio, gli indici per velocizzare le ricerche e la sicurezza.

Fasi della progettazione fisica

1. Scelta del DBMS e delle strutture di archiviazione

- Selezione del database management system (es. MySQL, PostgreSQL, SQL Server, Oracle).
- Configurazione dei file di database e delle strategie di memorizzazione.
- Definizione dei tipi di dati ottimali per ogni campo.

2. Definizione della struttura delle tabelle

- Ottimizzazione dei tipi di dati (es. INT, VARCHAR, TEXT, BLOB).
- Definizione di vincoli come NOT NULL, DEFAULT, CHECK.

3. Creazione di indici

- **Indice primario (Primary Index)**: basato sulla chiave primaria.
- **Indici secondari**: per velocizzare le ricerche su altri campi usati nei filtri (INDEX in SQL).
- **Indici full-text**: per ottimizzare ricerche testuali.

4. Gestione delle chiavi e vincoli

- Implementazione delle chiavi primarie e delle chiavi esterne con FOREIGN KEY.
- Definizione di politiche di ON DELETE CASCADE, ON UPDATE SET NULL, ecc.
- Gestione dei vincoli di unicità (UNIQUE).

5. Partizionamento e ottimizzazione dello storage

- **Partizionamento**: suddivisione di tabelle molto grandi in più parti per migliorare le performance.
- **Denormalizzazione** (quando necessario): creazione di ridondanze controllate per velocizzare le query.

6. Gestione della concorrenza e transazioni

- Implementazione di **lock** (es. READ COMMITTED, SERIALIZABLE).
- Configurazione delle strategie di isolamento delle transazioni.

7. Sicurezza e controllo degli accessi

- Creazione di utenti e ruoli con permessi specifici (GRANT, REVOKE in SQL).
- Implementazione della crittografia per i dati sensibili.

8. Backup e ripristino

- Definizione di strategie di backup (full, incrementale, differenziale).
- Configurazione di repliche per alta disponibilità.

Conclusioni

La progettazione di una base di dati è un passaggio fondamentale nella creazione di applicazioni che gestiscono informazioni. Una buona progettazione garantisce che i dati siano facilmente accessibili, efficienti e sicuri. In ogni fase della progettazione, è importante considerare le necessità specifiche dell'applicazione e scegliere le strutture più adatte per il contesto in cui si opererà.